Received 28 August 2023, accepted 28 September 2023, date of publication 11 October 2023, date of current version 13 November 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3323801

RESEARCH ARTICLE

Optimizing Reinforcement Learning-Based Visual Navigation for Resource-Constrained Devices

U. VIJETHA, (Graduate Student Member, IEEE), AND V. GEETHA, (Senior Member, IEEE)
National Institute of Technology Karnataka, Surathkal 575025, India

Corresponding author: U. Vijetha (vijethau.207it008@nitk.edu.in)

ABSTRACT Existing work on Deep reinforcement learning-based visual navigation mainly focuses on autonomous agents with ample power and compute resources. However, Reinforcement learning for visual navigation on resource-constrained devices remains an under-explored area of research, primarily due to challenges posed by processing high-dimensional visual inputs and making prompt decisions in real-time scenarios. To address these hurdles, we propose a State Abstraction Technique (SAT) designed to transform high-dimensional visual inputs into a compact representation, enabling simpler reinforcement learning agents to process the information and learn effective navigation policies. The abstract representation generated by SAT effortlessly serves as a versatile intermediary that bridges the gap between simulation and reality, enhancing the transferability of learned policies across various scenarios. Additionally, our reward shaping strategy uses the data provided by SAT to maintain a safe distance from obstacles, further improving the performance of navigation policies on resource-constrained devices. Our work opens up opportunities for navigation assistance and other applications in a variety of resource-constrained domains, where computational efficiency is crucial for practical deployment, such as guiding miniature agents on embedded devices or aiding visually impaired individuals through smartphone-integrated solutions. Evaluation of proposed approach on the AI2-Thor simulated environment demonstrates significant performance improvements over traditional state representations. The proposed method provides 84.18% fewer collisions, 28.96% fewer movement instructions and 11.3% higher rewards compared to the best alternative options available. Furthermore, we carefully account for real-world challenges by considering noise and motion blur during training, ensuring optimal performance during deployment on resource-constrained devices.

INDEX TERMS Collision detection and avoidance, reinforcement learning (RL), resource-constrained settings, reward shaping, Sim2Real transferability, state abstraction, visual navigation.

I. INTRODUCTION

Visual navigation has become a crucial aspect of autonomous systems, enabling a range of tasks including navigation, positioning, mapping, and path-planning. Reinforcement learning has emerged as a powerful approach for enabling intelligent agents to learn navigation policies directly from visual inputs without the need for explicit maps or pre-programmed instructions. However, deploying such reinforcement learning-based systems on resource-constrained devices presents unique challenges due to limited computational resources, memory constraints, and power limitations. This paper aims to propose an novel approach for enhancing the performance of reinforcement learning-based visual navigation tasks, explicitly designed to address the requirements of resource-constrained devices.

One of the key strategies that can be adopted to make reinforcement learning feasible on resource-constrained devices is dimensionality reduction. Our objective involves optimizing the computational and memory requirements of the reinforcement learning model by reducing the dimensionality of visual input data. This dimensionality reduction allows us to use the full potential of the model while overcoming the limitations of resource-constrained environments. To reduce dimensionality, researchers have explored techniques like image downsizing [1], graph representations [2],

The associate editor coordinating the review of this manuscript and approving it for publication was Jiachen Yang.

distance-based state representation [3], and latent representations [4], [5]. However, these techniques, while effective, still exhibit some degree of dependency on the training data. This dependence can impact the model's efficiency when applied to new and unseen environments. To address this dependency challenge associated with data dimensionality reduction, we introduce a State Abstraction Technique that abstracts the captured images into low-dimensional alternative representations that are independent of the training data. Moreover, smaller state spaces promote faster and more efficient learning in RL models.

While our primary focus has been on achieving dimensionality reduction for resource-constrained devices through the State Abstraction Technique, an interesting outcome of our approach is its seamless sim2real transferability. This transferability, crucial in bridging the gap between simulated and real-world environments, is often a significant challenge faced by most RL systems trained solely in simulation. Techniques such as domain randomization [6], domain adaptation and policy distillation [7] have been introduced in literature to address this issue. Although these techniques are effective in addressing sim2real transferability challenges, they often require substantial effort in data collection, fine-tuning, and parameter tuning. In contrast, our method inherently captures the sim2real transferability feature, allowing the reinforcement learning model to generalize effectively to real-world scenarios without the need for explicit domain adaptation or extensive domain randomization. This seamless transition from simulation to reality, combined with the advantages of dimensionality reduction for resource-constrained devices, highlights the robustness and practicality of our proposed approach in real-world visual navigation applications.

To further enhance the adaptability of our approach to real-world scenarios, we address the challenge of reward modeling in visual navigation tasks. Typically, agents in such tasks rely on distance measurement units, which are essential for constructing reward models that penalize the agents for approaching or colliding with obstacles and reward them for successful target achievement [8], [9], [10], [11], [12], [13]. However, it's crucial to note that without access to distance measurement units, these reward modeling strategies become ineffective. In response, our proposed method offers a solution by using information from the abstracted state to estimate distances to nearby obstacles. This innovative approach enables effective distance-based reward modeling even in scenarios where explicit distance measurements are unavailable. Yet, another significant challenge in this domain is the sparsity of rewards. Given that collisions with obstacles are infrequent occurrences in visual navigation, agents often find themselves engaged in aimless exploration during training, which hinders their ability to gain meaningful task-related insights. To address this problem, the proposed method uses reward functions that bestows agent with smaller but more frequent rewards for demonstrating collision-avoidance behaviors, thereby significantly improving their learning efficiency.

In visual navigation, captured images often contain noise and redundancy, presenting challenges in extracting relevant information for effective navigation. As a response, researchers have explored alternative inputs such as depth images and semantic segmentation maps. These alternate formats reduce the dimensionality of the data while preserving the important details about the image. The proposed system for visual navigation uses semantic segmentation images of the captured frames, that provide high-level information about objects and their spatial locations in the scene while reducing noise and redundancy compared to RGB images. We introduce the State Abstraction Technique as a method to condense semantic segmentation images into a more compact format. This approach not only boosts the learning process of the RL policy network but also enhances the overall performance of the navigation system. By using SAT, we aim to address the limitations of high dimensionality while enhancing the transferability of learned policies from simulation to real-world environments. We summarize our contribution as follows.

- We propose an innovative State Abstraction Technique that simplifies complex observations of dynamic environments into a simple, intuitive format, making it compatible with any reinforcement learning-based visual navigation task.
- We introduce a novel Reward Shaping module that uses the extracted state information to enforce collision avoidance and safe navigation behaviors in a Reinforcement Learning agent.
- We thoroughly analyze different parameters to understand their impact on the effectiveness of the proposed SAT approach, providing insights for fine-tuning and optimization to achieve optimal results in real-world scenarios.
- We deploy and test our proposed system as a smartphone app for navigation in both indoor and outdoor environments, highlighting its real-world applicability and potential for practical usage in resource-constrained settings.

The remainder of the paper is structured as follows. Section II introduces the Related Work in Reinforcement Learning based Visual Navigation. The Proposed Methodology is described in detail in Section III. We discuss the details of the important Design Decisions made in the proposed work in section IV. Section V describes Training and Experimentation in detail. The Results and Analysis of the proposed system performance are presented in section VI. Finally section VII concludes the paper.

II. RELATED WORK

The Related Work section provides an overview of existing research about Reinforcement learning-based Visual Navigation. The section covers three principal areas of focus. Firstly, we explore the reward modeling approaches employed by various reinforcement learning techniques to effectively tackle collision avoidance and target-reaching

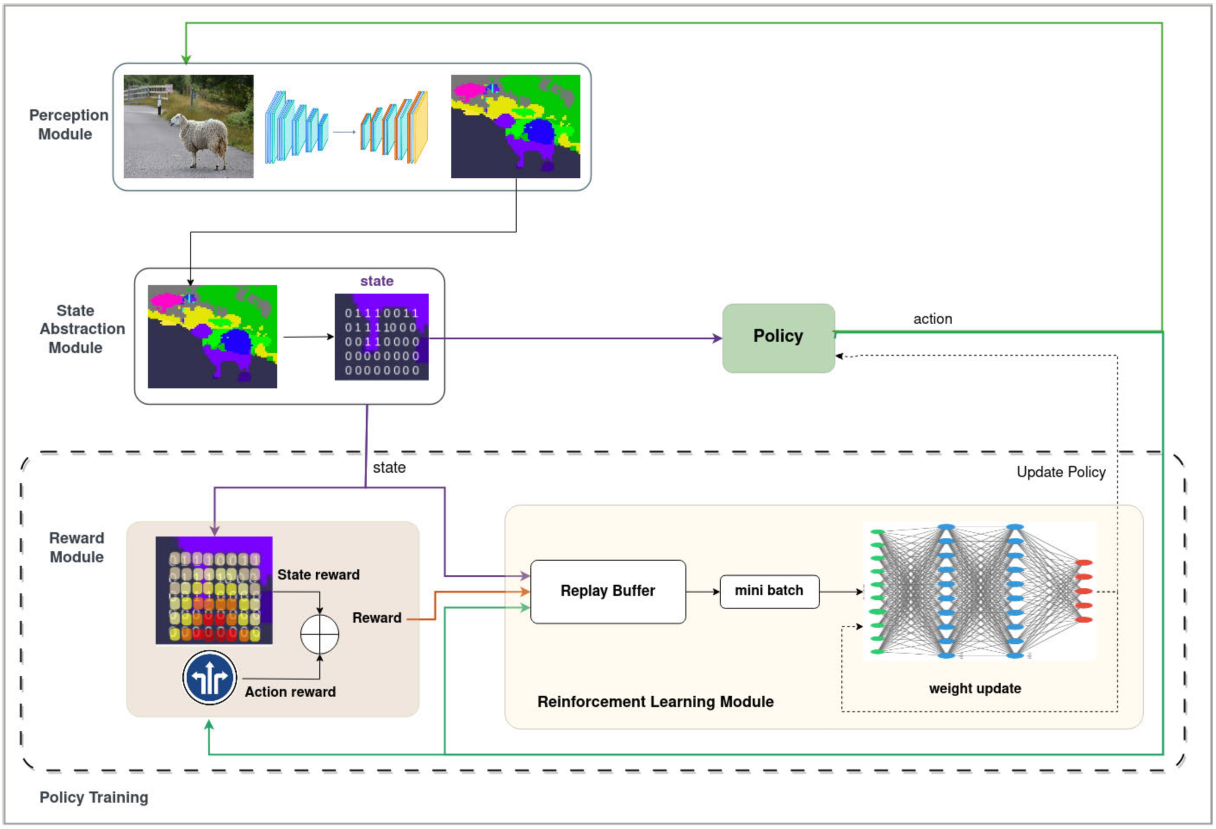

FIGURE 1. **Proposed system for navigation guidance.**

tasks. Additionally, we explore the efforts made to address the challenges posed by high-dimensional visual inputs in reinforcement learning. Furthermore, we discuss the existing work on Sim2Real transferability, which aims to bridge the gap between simulated and real-world environments.

A. REWARD MODELING IN AUTONOMOUS NAVIGATION

Reinforcement learning uses a system of rewards and penalties to encourage desired behavior and discourage harmful behavior. Accordingly, approaches in obstacle avoidance [8], [9], [10], [11], [12], [13] assign a penalty to agents when it goes near an obstacle or collides with an obstacle, indicating that the behavior is not desirable. Ye et al. [14] employ supervised learning and RL to learn the membership functions of fuzzy rules for the task of obstacle detection. These rules are then used to classify objects as obstacles. Goal-based navigation agents [15], [16], [17], [18] are more focused on reaching the goal and with the shortest trajectory. In addition to the penalty issued for collision with obstacles, these agents are generously rewarded for reaching their goal. To encourage shorter trajectories, the penalty issued to the agent is increased as it moves away from the goal and reduced as it moves toward the goal point. Reinforcement learning is used to enforce a desirable behavior to ensure that the agent is penalized when the behavior deviates from the expected behavior. Huang et al. [4] encourage optimal driving behavior by rewarding/penalizing the agent based on on-road/off-road traits and for maintaining a safe driving speed.

The current approaches to reward modelling for collision avoidance mainly depend on distance measurement mechanisms, involving the continuous monitoring of both agent and object positions as well as their orientations. However, when considering scenarios involving navigation assistance that exclusively relies on visual inputs, the conventional methods prove to be inadequate. In response to this limitation, we introduce the State Abstraction Technique, which extracts comprehensive state information from the environment. This technique provides useful insights into the positions of objects and their likely distance approximations, all derived exclusively from visual information.

B. DIMENSIONALITY REDUCTION

To deal with high dimensional inputs in visual navigation, Raudies et al. [3] compute distances to objects in multiple directions using optical flow or stereo vision, and replace visual images with these distances as the state information. Cornel et al. [19] transform visual inputs into an abstract tabular model to reduce the state space. Eysenbach et al. [2] represent observations and distances as nodes and edges of a graph to find the shortest trajectory, in an attempt to reduce the dimensionality of the state space. Huang et al. [20] fuse information from multiple cameras to compensate for partial observation from a single monocular camera and then use a Borderline network and attention network to reduce the state space and retain important information only. Liu et al. [21] also use temporal attention to reduce the dimension of state-action pairs. Huang et al. [4] propose a semantic encoder module (SEM) to extract low-dimensional driving representations from raw image observations. Several attempts have been made to reduce the dimensionality of visual inputs in resource-constrained settings. Schoettler et al. [1] scale down the visual inputs to 32×32 grayscale image to accommodate the requirements of industrial Insertion tasks. Shah et al. [5] use low dimensional ResNet features as compact representations that capture salient features of visual inputs. Zhu et al. [22] employ Variational Autoencoders to convert real-world visual inputs into corresponding low dimensional latent representations that can be easily processed by RL models.

Existing techniques rely on methods to generate reduced-dimensional representations of the given training data. This facilitates efficient training of reinforcement learning models. However, these methods tend to retain some dependency on the training data causing the models' performance to drop when confronted with novel and unfamiliar scenarios. In contrast, the proposed method performs 2 levels of abstraction on training data resulting in compact state representations with a remarkable autonomy from the specifics of the training environment. As such, the model that undergoes rigorous training on a small set of scenarios gains an exceptional level of adaptability. This enables the model to seamlessly adapt to completely novel and previously unseen virtual and real-world environments.

C. SIM2REAL TRANSFERABILITY

Reinforcement learning models are typically trained using historical experience data or in a simulated environment, after which the knowledge is applied to similar real-world situations. However, training the agent in a specific environment increases the model's reliance on the training environment and reduces its transferability, resulting in mediocre performance in new and unseen scenarios. To reduce the dependency of the model on the training environment, Wu et al. [23] captures prior knowledge of the scenes from the training environment and represent the relation between the locations inside the scene as a Probabilistic Relational Graph. This abstract knowledge about the environment could be easily transferred to similar environments in virtual and real environments. Devo et al. [24] attempt Sim2Real transferability by assigning the task of learning target-driven visual navigation to 2 separate networks, one that attempts to locate the target, and another learns to explore the environment to move towards the target. By breaking down the task into simpler tasks, the authors attempt to increase the generalization ability of the model in new environments. Policy distillation techniques [7] extract the policy of a reinforcement learning agent and train a new network that performs at the expert level, while being smaller and more efficient. This technique consolidates multiple task-specific policies into a single policy. Domain randomization [6] involves training the agent in a variety of simulated environments that differ in many ways (e.g., lighting, textures, object appearance) to help it learn more robust policies that can generalize to real-world environments. Semantic segmentation is often used as an intermediary format to effectively bridge the gap between simulation and reality [4], [25], [26]. The use of semantic segmentation can also help to reduce the dimensionality of the input space, resulting in more efficient and effective model training. However, the variations and inconsistencies in the real world may not always be fully captured by the simulated environment used for training. Therefore, although semantic segmentation can be used to bridge the gap between virtual and real environments, it does not solve the Sim2Real problem entirely.

Policy distillation, domain randomization, and domain adaptation require fine-tuning the model trained within a simulated environment when transitioning to real-world environments. However, even with this fine-tuning, the model might not achieve complete transferability to entirely new environments due to its inherent reliance on the specifics of the training environment and its variations. The two-level abstraction introduced in the proposed method addresses this challenge by enabling the model to seamlessly transition to novel environments without the need of any additional fine-tuning.

The proposed system aims to address the challenges of high dimensionality and Sim2Real transferability in reinforcement learning-based visual navigation through State Abstraction Technique. This is particularly relevant in resource-constrained settings, where computational resources are limited. The goal is to reduce the dependency of state information on the training environment, making the extracted state applicable to any scenario in both virtual and real environments. By focusing on resource-efficient strategies, our approach offers a promising solution for enhancing navigation performance in resource constrained settings.

III. METHODOLOGY

Visual navigation is the task of analyzing visual inputs to find and direct an agent along a walkable, obstacle-free path in an environment. Traditional approaches rely on depth [27], [28], [29], [30] and semantic segmentation [31], [32], [33] data for estimating obstacle distances and road profiles, but they are limited in their transferability to new domains due to their dependence on specific datasets.

To address these challenges, we present an RL-based visual navigation system that uses a novel State Abstraction Technique that extracts a compact navigation-specific state information from the environment observations. This extracted state information not only reduces dimensionality but also enhances the system's capability for seamless

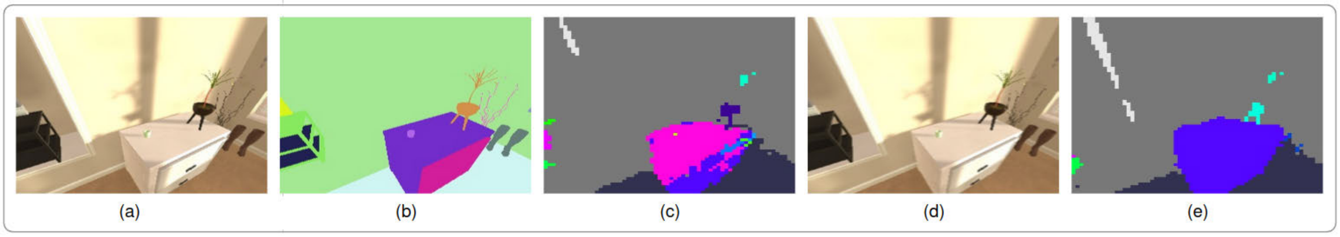

FIGURE 2. Semantic segmentation of the agent captured images. (a)Captured image, (b) Semantic segmentation image provided by the framework, (c) Semantic segmentation image externally generated by a custom model, (d)Motion blur on the Captured image, and (e)Semantic segmentation of blurred image.

transfer between simulated and real-world environments. Additionally, it effectively highlights obstacle positions and distances, serving as a valuable indicator for successful navigation. The proposed system architecture comprises four main components: a Perception module that extracts useful information from images captured by an agent-mounted camera, a SAT module that transforms this information into abstract state representations, a Reward module that enforces Obstacle avoidance behaviors of the agent in complex environments, and an RL module that learns a policy for safe navigation. Fig. 1 illustrates the architecture of the proposed system. In this setup, the agent observation goes through 2-levels of abstraction: (i) Conversion of RGB images to semantic segments in Perception module, and (ii) Conversion of semantic segments into compact state information in the SAT module. The Policy model receives this abstract state representation as input and produces discrete, position-based motion primitives as actions (e.g., move ahead, move left or right, turn left or right). The proposed system, being an online training setup, records the details of each agent interaction in a Replay buffer and uses these historic observations to train the Policy network.

The system uses a State Abstraction Technique to generate an abstract, compact representation of the agent observations. The state derived through SAT effectively tackles the challenge of high-dimensional visual inputs and enhances sim2real transferability. Moreover, this extracted state serves as input to the reward-shaping module, contributing to the formulation of efficient obstacle-avoidance strategies. The components of the proposed work together to enable the safe navigation of the agent in complex environments. Detailed information on each component of the proposed system is provided in the subsequent subsections.

A. PERCEPTION MODULE

Visual navigation involves using information in captured images to perform navigation tasks. However, raw RGB images can be overly detailed and redundant, leading to a large state space. Larger state spaces demand a larger number of interactions with the environment to learn an efficient control policy.

To resolve this issue, smaller alternative representations, such as depth maps and semantic segmentation maps are often used for visual navigation tasks. Depth maps contain information about the distance between objects in the scene and can be used for obstacle avoidance and navigation guidance tasks [34], [35], [36]. However, depth data only provides distance information and discards other important information that could otherwise be used to identify objects for a better navigation experience. In contrast, semantic segmentation maps provide a high-level representation of the environment that can aid navigation by providing information about the locations of objects and landmarks.

We chose semantic segmentation over depth representation as it provides a more detailed and comprehensive understanding of the environment, enabling better navigation decisions. The success of Sim2Real transferability of semantic segmentation images [4], [25], [26], [34] serves as an additional rationale for the selection of this representation for our navigation guidance task.

The proposed system is trained for Navigation Guidance in the simulated environment of the AI2-Thor framework [37], which provides a fully functional 3D interactive environment with access to semantic segmentation images at each interaction. However, when deploying the proposed system in the real world, a model is required for generating semantic segmentation images from the captured images. It should be noted that the quality of semantic segments obtained from the virtual environment may contain more detailed and refined information, which may not align with the semantic segmentation images generated by state-of-the-art deep learning models for semantic segmentation, as illustrated in Fig. 2.

Moreover, the images captured by the agent in a simulated environment are often visually appealing and do not capture the imperfections of the real world, such as noise and motion blurriness. To make the model adapt to real-world conditions, we introduce motion blur to the captured images to reflect the images captured by a moving agent in real-world scenarios. This can be represented as a convolution of the input image I with a motion blur kernel $B(x, y)$, resulting in the blurred image $I_{blurred}(x, y)$ as shown in (1).

$$I_{\text{blurred}}(x, y) = \sum_{k=-K}^{K} \sum_{l=-L}^{L} I(x + k, y + l) \cdot B(k, l) \qquad (1)$$

Algorithm 1 Computation of State Space

```
1:  $m, n \leftarrow$ predefined number of rows and columns for state space
2:  $I_s \leftarrow$ Semantic Segmentation of agent observation I
3:  $StateGrid \leftarrow$ State space grid around agent position $(agent_x, agent_y)$
4:  $StateMatrix \leftarrow$ Array(dimension:$(m, n)$, Values: 0)
5:  for each $grid \in StateGrid$ do
6:      $segments_{grid} \leftarrow$ getSegmentClasses($grid$)
7:      $\text{segment}_{\text{max\_strength}} \leftarrow$ getSegmentWithMaxStrength($segments_{\text{grid}}$)
8:      if $segment_{max\_strength} \notin \langle$path segment$\rangle$ then
9:          $StateMatrix[grid_{\text{row}}][grid_{\text{col}}] \leftarrow 1$
10:     end if
11: end for
```

where K and L are the dimensions of the motion blur kernel, $I(x, y)$ is the intensity value of the input image at pixel coordinates (x, y), and $B(k, l)$ is the intensity value of the motion blur kernel at kernel coordinates (k, l).

Fig. 2 illustrates the semantic segmentation images obtained from the captured image and the image with motion blur. It can be observed that that the semantic segmentation generated from the blurred image (Fig. 2(e)) has inferior quality compared to the framework-supplied images as well as actual images, as shown in Fig. 2(b&c). However, despite the reduced quality, introducing motion blur to the captured images brings the agent one step closer to achieving Sim2Real transferability. This approach helps the model to better adapt to real-world conditions where images captured by a moving agent may contain noise and motion blurriness. The resulting semantic segmentation image with motion blur is used in the subsequent steps of the processing pipeline, enabling the proposed system to better generalize to real-world scenarios.

Any efficient state-of-the-art real-time semantic segmentation model can be used in the Perception module. We use the Topformer [38] model pretrained on ADE20k dataset in the proposed system. TopFormer has achieved state-of-the-art performance in benchmark datasets, capturing fine-grained object boundaries, handling complex scenes, and showing promising transferability. With its high-quality segmentation maps produced in real time, TopFormer provides a detailed understanding of the environment, making it a promising choice for accurate scene understanding in navigation applications.

B. PROPOSED STATE ABSTRACTION TECHNIQUE (SAT)

The Semantic segmentation obtained from the perception module can still pose a large state space for RL models. A bigger state space requires the agent to have a large number of interactions with the environment to cover all possible state-action combinations. To address the state space problem, solutions such as down-scaling images [1], extracting ResNet features [5], generation of latent representation using Autoencoders [4], [22] have been discussed in literature.

Although effective in reducing the state space, these techniques keep the model dependent on the observations

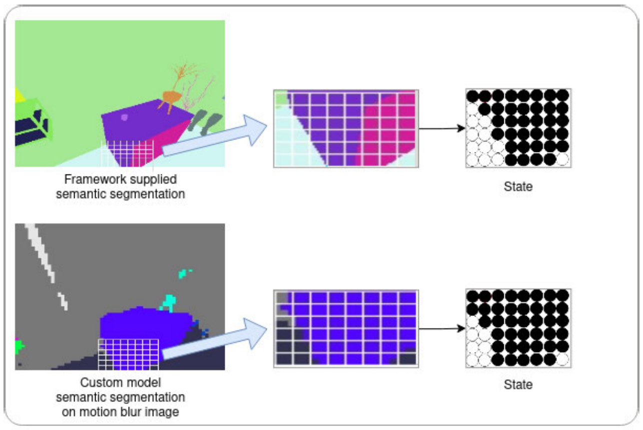

FIGURE 3. Process of extraction of State information Black - obstacle zone, White - obstacle free zone.

from the training environment and limit the generalization capability of the model to new, unseen environments. To address this challenge, we propose a State Abstraction Technique that extracts a compact, abstract representation of the environment observation O that is independent of the training environment. The SAT can be represented as a function $f_{\text{SAT}} : O \rightarrow \{0, 1\}^{m \times n}$, where m and n are the dimensions of the state space. The SAT is implemented in the State Abstraction Module shown in Fig. 1.

Given a semantic segmentation image, the entire image content may not be relevant to the task of visual navigation. Instead, a portion of the image that represents the immediate neighborhood of the agent serves as the region of interest as shown in Fig. 3. To define the state space, the region of interest in the observation is divided into a grid of cells. Each cell is assigned a binary value based on whether it contains a path segment (road, floor, earth, sidewalk, ground) or an obstacle segment. The state matrix S is then defined as a binary matrix of size $m \times n$, where

$$S_{i,j} = \begin{cases} 1 & \textit{if}\,\text{(contains obstacle)} \\ 0 & \textit{Otherwise} \end{cases} \tag{2}$$

The proposed approach significantly reduces the Observation space while still capturing the relevant information for the task at hand. Algorithm 1 outlines the process of

Algorithm 2 Reinforcement Learning Module

```
Require: Replay buffer $\mathcal{D}$ with capacity $N$, Q-network parameters $\theta$, Target network parameters $\theta'$, Discount factor $\gamma$,
    Minibatch size $B$, Learning rate $\alpha$, Number of iterations $T$
 1: Initialize Q-network with parameters $\theta$
 2: Initialize target network with parameters $\theta' \leftarrow \theta$
 3: for $t = 1$ to $T$ do
 4:     Observe state $s_t$
 5:     With probability $\epsilon$ select a random action $a_t$, otherwise select $a_t = \arg\max_a Q(s_t, a; \theta)$
 6:     Execute action $a_t$ and observe reward $r_t$ and next state $s_{t+1}$
 7:     Store transition $(s_t, a_t, r_t, s_{t+1})$ in $\mathcal{D}$
 8:     Sample minibatch $\mathcal{B}$ of size $B$ from $\mathcal{D}$
 9:     Set target for minibatch transition $(s, a, r, s')$ as $y = r + \gamma \max_{a'} Q(s', a'; \theta')$
10:     Update Q-network parameters using gradient descent on loss $\mathcal{L}(\theta) = \frac{1}{B}\sum_{(s,a,r,s')\in\mathcal{B}}(y - Q(s, a; \theta))^2$
11:     Every $C$ steps, update target network parameters $\theta' \leftarrow \theta$
12: end for
```

generation of state information from the given semantic segmentation image.

Fig. 3 highlights the significance of the abstraction process in reducing the gap between the refined observations of the simulated environment and the noisy blurred images of the real-world conditions. It can be observed that, although the semantic segmentation of the blurred image is noisy and inconsistent, the state information produced by it is a good approximation of the state produced by the semantic segmentation image provided by the simulation environment.

C. PROPOSED REWARD MODULE

Navigation guidance in reinforcement learning requires a reward system that provides feedback to the agent about its actions in a given state. A reward is typically used to encourage desirable behavior, while a penalty is used to discourage undesirable behavior. The desirable behaviour for collision avoidance in visual navigation can be enforced by penalising the agent whenever it collides with an obstacle. During environment interactions, the distribution of such events of collision with obstacles can be quite sparse, depriving the agent of ample opportunities for learning efficient policies. A sparse reward system with a penalty for collision and a reward for reaching the destination may not be effective in teaching the agent to avoid obstacles. We need a dense reward/penalty model that enforces the agent to stay away from obstacles. Wenzel et al. [34] use the distance to the obstacle in their reward function design to enforce a policy that keeps the robot as far away as possible from the obstacle. Huang et al. [4] set a reward system that encourages the vehicle to drive as close as possible to the middle of the road. However, in our context of pure visual navigation, measuring the distance to the obstacle is not feasible. Instead, we use the state information produced by the State model as an alternative for proximity estimation.

During the training process, we adopt a reward shaping technique inspired using the concept of repulsive potential field to guide the agent's behavior in avoiding obstacles. The agent is rewarded for occupying obstacle-free walkable states, while it is penalized as it moves towards obstacles. To enforce the desired behavior of moving away from obstacles, we designate a small neighborhood around the agent as a penalty zone. This penalty zone represents an immediate collision risk to the agent and should be maintained as an obstacle-free region.

To create a repulsive effect that encourages the agent to stay away from obstacles within the grid-based neighborhood, we use a *PenaltyMatrix*, that represents the repulsive potential values corresponding to the agent's proximity to obstacles, with the repulsive potential diminishing as the distance increases. The Repulsive potential R_g at each position of the grid is computed as shown in (3).

$$R_g(S_{i,j}) = S_{i,j} * PenaltyMatrix[i][j] \tag{3}$$

where the StateMatrix value $S_{i,j}$ represents the presence or absence of obstacle in the corresponding location in the grid and $PenaltyMatrix_{i,j}$ represents the repulsive potential value associated with that location in the grid.

The navigation instruction set supported by the agent are MoveLeft, MoveAhead, MoveRight, TurnLeft, and TurnRight. A desirable action a for the agent would be to lead along a safe, walkable path with the fewest navigational instructions possible. To discourage unnecessary navigation instructions, a penalty R_a is served for certain actions as shown in (4). Unnecessary issue of changing directions using TurnLeft or TurnRight is avoided by assigning higher penalty to those actions. This forces the agent to issue a TurnLeft or TurnRight command only when absolutely necessary. To encourage the agent to identify straight walkable path, the agent is rewarded with a $+R_{ahead}$ for every MoveAhead instruction.

$$R_a(S_t) = \begin{cases} -R_{move} & a \in (\text{MoveLeft, MoveRight}) \\ -R_{turn} & a \in (\text{TurnLeft, TurnRight}) \\ +R_{ahead} & Otherwise \end{cases} \tag{4}$$

The cumulative reward R_c across the state space can be computed as shown in (5).

$$R_c(S_t) = (\sum_{i=0}^{n} \sum_{j=0}^{m} R_g(S_{i,j})) + R_a(S_t) \tag{5}$$

The reward R_g serves as an indicator of the proximity of the agents to obstacles in the environment. The cumulative reward R_c incorporates both this proximity information and the desirable action executed by the agent. When the agent successfully adjusts its path to avoid obstacles, resulting in $R_g = 0$, a bonus reward R_{safe} is granted. Otherwise, the agent receives $R_{safe} - R_c$ points. The final reward computation is summarized in equation (6), where the reward decreases as the agent approaches the obstacle and increases as it moves away.

$$reward = R_{safe} - R_c \tag{6}$$

D. REINFORCEMENT LEARNING MODEL

Vision-based navigation is a decision-making task in which an agent learns to interact with its environment based on feedback. The goal is to predict the desirable action for each observation, with the intent of keeping the agent in a safe state. This can be formulated as a Markov decision process (MDP), which is defined by a tuple (S, A, π, R, γ), where S is the observation space, A is the action space, π is the policy function used for state transition, R is the reward function, and γ is a discount factor. The state space and reward data are obtained from a State Abstraction Module and Reward Module respectively.

1) ACTION SPACE

Five actions are defined in the action space for the navigation of the agent namely: MoveLeft, MoveAhead, MoveRight, TurnLeft, and TurnRight. Each action corresponds to a motion command to be performed by the agent. The MoveLeft and MoveRight actions inform the agent to shift a bit to the left or right, respectively, as an attempt to avoid collision with an immediate obstacle. The agent issues a TurnLeft or TurnRight command when the path ahead is blocked and the agent has to change direction. The MoveAhead action instructs the agent to keep walking in the current direction.

The learning process begins with exploration and gradually progresses towards exploitation. Initially, the agent explores every state-action pair in the environment by performing random actions and learning without regard for the current state. Eventually, the agent progresses to exploitation, in which it employs the learned knowledge to guide the choice of actions that maximise the reward of the current state. The ϵ-Greedy is a hyperparameter used to balance exploration and exploitation by choosing between exploration and exploitation randomly.

2) POLICY FUNCTION

The task of the agent is to learn a policy, $\pi : S \rightarrow A$, for selecting the next action a_t, based on the current observed state s_t; i.e. $\pi(s_t) = a_t$. The final reward assigned to the agent serves as the driving force for the state transition function. A higher reward encourages the agent to remain in the same state, whereas a lower reward or penalty encourages the agent to transition to a more desirable state. The agent's optimum course of action is to adopt a policy that maximizes its cumulative reward in a particular environment, and one of the key factors in determining this is the state transition function.

Algorithm 2 highlights the working of RL model for navigation guidance. To enhance the navigation experience, it is desirable for a solution not only to maximize episodic reward but also to minimize the number of navigation instructions required. This can be achieved through careful consideration of the exploration-exploitation tradeoff. Specifically, the ϵ-greedy strategy can be employed to encourage the agent to initially explore all possible actions, while gradually learning that certain actions, such as 'Move Ahead', are more beneficial for successful navigation. By doing so, the agent can learn an effective policy that achieves the desired behavior with minimal instruction requirements.

E. NETWORK ARCHITECTURE

The Double Deep Q-Network (DDQN) algorithm uses two neural networks: the primary Q-network and the target Q-network. The primary Q-network is updated every iteration, and it is used to choose actions and evaluate their Q-values. The target Q-network is used to calculate the target Q-values, which are used to update the primary Q-network. The target Q-network is updated less frequently than the primary Q-network, and it is used to provide a stable target for the Q-value updates.

The update equation for the primary Q-network is:

$$\begin{aligned} Q(s_t, a_t) \leftarrow\ & Q(s_t, a_t) \\ & + \alpha(r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)) \end{aligned} \tag{7}$$

where s_t and a_t are the state and action at time step t, r_{t+1} is the reward received after taking action a_t in state s_t, α is the learning rate, and γ is the discount factor.

The update equation for the target Q-network is:

$$\begin{aligned} Q_{target}(s_t, a_t) \leftarrow\ & Q_{target}(s_t, a_t) \\ & + \beta(r_{t+1} + \gamma \max_a Q_{primary}(s_{t+1}, a) \\ & - Q_{target}(s_t, a_t)) \end{aligned} \tag{8}$$

where β is the update rate for the target Q-network, and Q_{primary} is the output of the primary Q-network.

1) NETWORK STRUCTURE

Each network consists of three fully connected layers with 64, 64, and 5 (action space) neurons respectively. The input to the network is a state vector of size $m \times n$ representing the current state space of the agent. The first four layers of

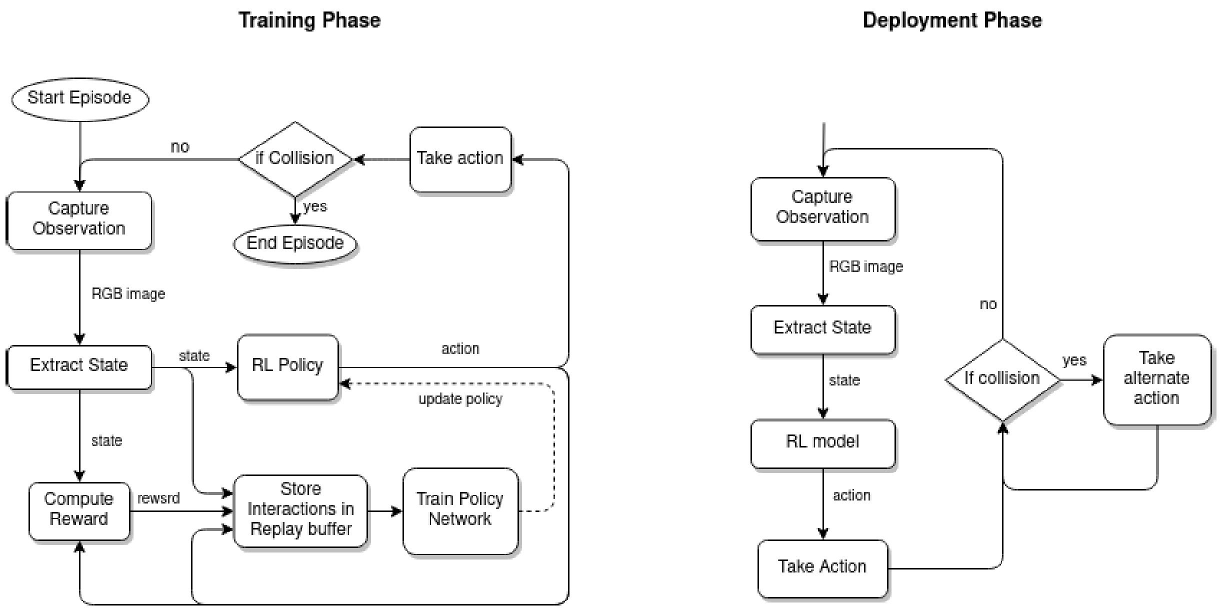

FIGURE 4. System workflow in training and deployment phases.

the network use the rectified linear unit (ReLU) activation function, while the last layer uses a linear activation function to produce the Q-values for each possible action. The network uses the Adam optimizer with a learning rate of 0.001 to update the weights.

2) HYPERPARAMETERS

The DDQN algorithm involves several key settings that influence training. Training is performed using batches of 64 samples, leading to network updates after every 64 samples. The discount factor for calculating target Q-values is set at 0.99, influencing the balance between future and immediate rewards. Initially, exploration is encouraged with an exploration rate of 1, which progressively reduces linearly to 0.01 across 1 million steps. The target network is updated every C steps, controlling how often target Q-values are synchronized with current Q-values. These hyperparameters, determined through prior experiments, aim to strike a balance between exploration and exploitation and stabilize the learning process.

F. TRAINING AND DEPLOYMENT PHASES

Fig. 4 provides an overview of the workflow for the proposed system during both the training and deployment phases. In the training phase, the agent captures environmental observations in the form of monocular RGB images. Using the previously discussed SAT technique, relevant state information is extracted from these observations. Subsequently, this state information is fed into the RL policy network to predict an appropriate action. In cases where the agent's action leads to a collision within the environment, the episode is promptly terminated, incurring a significant penalty to indicate an undesirable event. In the absence of a collision, the agent continues to explore the environment, further training the policy network. During each interaction, the state information and the predicted action are used to compute rewards for the policy network's training. All relevant details of each agent-environment interaction are stored in a replay buffer, from which historical information is sampled and used for training the RL Policy Network.

The deployment phase is rather straightforward, making use of the fully trained RL model for navigation decisions. In this phase, the agent continues to capture environmental observations as monocular RGB images. The state information, extracted from the preprocessed observations using the SAT technique, is then provided to the RL model to generate an appropriate action for the current context. The RL model outputs a probability distribution of the five possible actions. The action with the highest probability is selected and executed. In the event that this chosen action results in a collision with the environment, an alternative action, typically the next best action based on the RL model's results, is chosen and executed iteratively until collision avoidance is achieved. The agent continues to navigate the environment by capturing observations and determining suitable actions for each new observation. The primary objective of the proposed method is to design and train a Reinforcement Learning (RL) model capable of minimizing the frequency of collisions within the environment while ensuring a seamless and smooth navigation experience.

IV. DESIGN CONSIDERATIONS

The performance of the proposed system is influenced by certain design decisions. We have shortlisted the factors of influence into 4 important criteria: (i)online/offline training (ii) the reinforcement learning technique used (iii)selection of state space size and, (iv) selection of grid size. The significance of these criteria and the details on design decisions made are explained in the following sections.

A. ONLINE VS OFFLINE TRAINING

A design choice has to be made whether neural networks are trained based on previously collected data (Offline training) or using additional data collected during the learning process (Online training). For large state spaces like the input images in visual navigation, providing an exhaustive list of historical data that covers all possible state spaces is not practical. Online training provide opportunities for the agent to explore the state space by continuous interaction with the environment. However, facilitating rigorous online training in real environments can be a challenging task. Hence most algorithms resort to simulated environments for environment exploration. The proposed system uses Online training on the simulated environment provided by AI2-Thor. The AI2-Thor simulation environment provide opportunities for the agent to move around indoor spaces, collect new training tuple from each interaction with the environment and develop obstacle-avoidance expertise.

B. SELECTION OF REINFORCEMENT LEARNING TECHNIQUE

Traditional reinforcement learning algorithms like Q-learning maintains a reference table mapping of states to actions. However, this approach becomes less effective in scenarios with dynamic environments where the model struggles to respond to unseen or unfamiliar observations. Recent techniques have replaced traditional 2D arrays with neural networks to estimate the Q-value function. Neural networks are robust to new inputs when compared to static 2D arrays and hence produce reliable results. To determine an optimal RL technique for the collision detection and avoidance task, we experiment with Deep Q Network (DQN), Double Deep Q Networks (DDQN), and Deep Deterministic Policy Gradient (DDPG). These neural network-based models are trained on a subset of the samples(state, reward, action, next state) recorded from the agent's interaction with the environment.

- DQN (Deep Q-Network) uses a policy network to learn an optimal policy and a target network to stabilize the learning process by periodically updating the parameters of the target network with the policy network's parameters.
- DDQN (Double Deep Q-Network) is an extension of DQN that addresses the overestimation of action values problem in traditional DQN networks. DDQN uses two separate Q-value estimators, producing more stable and reliable results.
- DDPG (Deep Deterministic Policy Gradient) is an actor-critic algorithm that works well for tasks with continuous action spaces. DDPG has two networks: an actor-network that selects the best action for a given state, and a critic-network that evaluates the policy function estimated by the actor-network.

Fig. 7 (a) shows the performance of the models on different scenes of AI2-Thor environment. An ideal method for navigation guidance would provide efficient navigation guidance with minimal collisions and fewer navigation instructions. Higher rewards are given to the agent when it diligently follows a safe, obstacle free path. Due to the discrete nature of the action space, the DQN and DDQN algorithms tend to outperform the DDPG algorithm in this task. DDQN outperforms the other techniques in terms of fewer collisions and movements and higher rewards.

C. STATE SPACE SIZE

The state space refers to the area surrounding the agent that is observed in order to navigate successfully. Choosing an appropriate size to represent the state space is crucial for achieving optimal navigation guidance. Increasing the size of the state space results in a larger observation area around the user, which in turn can improve navigation performance. To evaluate the impact of state space size on navigation performance, an experiment was conducted on 20 scenes in the Ai2-thor environment using models trained with state space sizes ranging from 4 to 11. The experiment recorded and averaged the number of collisions encountered, movement commands issued, and rewards earned by the agent for each state space size. The details of the analysis are shown in Fig. 7(b).

Contrary to the expectation that bigger state spaces yield better navigation experience, the experiments revealed that larger state sizes resulted in increased agent movement in an attempt to avoid obstacles in the larger observation zone, rather than improving navigation. The number of navigation instructions is also a significant factor. A balance between identifying potential obstacles and minimizing the number of navigation instructions is necessary to provide effective guidance. Fig. 7(b) illustrates the trade-off between the number of collisions and the number of navigation instructions. It can be observed that although state space of size 10 and 11 achieve few collision and better rewards, they also tend to have more navigational instructions issued which is undesirable for a smooth navigation experience. Experiments conducted in various environments using different state space sizes indicate that a state space of size 6 provides optimal results by identifying potential obstacles and minimizing navigation instructions.

D. GRID SIZE

The grid size of the state space refers to the size of each grid cell of the state space. The grid cell's state value is determined by its content. Therefore, selecting an optimal grid size is crucial for the success of the model. Overly fine-grained grids only inflate the size of the state space, adding little to system performance. Conversely, the excessive details

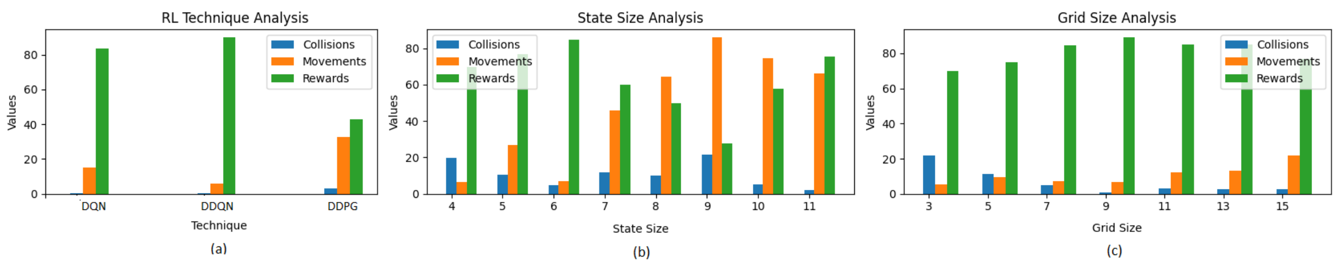

FIGURE 5. Effect of different design decisions on system performance.

contained in a larger grid may result in incorrect state values being computed for the cells. To evaluate the effect of grid size on performance, a series of experiments were conducted on 20 distinct scenes within the Ai2-thor environment using the DDQN model, where the state space size was set to 6 and grid sizes ranged from 3 to 15. The results of the experiments are summarized in Fig. 7(c), which outlines the performance metrics for each trial using different grid sizes. The findings demonstrate that a grid size of 9 achieved the optimal performance, with fewer collisions and navigation instructions.

V. EXPERIMENTAL SETUP

In this section we describe the dataset, the training setup, and the details of evaluation metrics and baseline techniques used for comparison with proposed method.

A. DATASET

The AI2-Thor framework [37], an interactive 3D virtual environment is used as a simulation environment for training and testing the proposed system. AI2-Thor framework consists of near photo-realistic 3D indoor scenes, where AI agents can navigate in the scenes and interact with objects to perform tasks. The framework provides 200+ custom built high-quality scenes, 2600+ custom designed household objects across 100+ object types. Each object is heavily annotated and allows for near-realistic physics interaction. The framework provides Multi-agent support, 200+ actions for a wide range of interaction and navigation based embodied AI tasks.It also provides support for many image modalities(ego-centric RGB images, instance segmentation, semantic segmentation, depth frames, normals frames, top-down frames, orthographic projections, and third-person camera frames) and camera adjustments (changing image size and field of view). After each step in the environment, there is a large amount of sensory data available about the state of the environment. This information can be used to build highly complex custom reward functions.

B. ADDRESSING STATE IMBALANCE IN TRAINING DATASET

In both real and virtual environments, certain states are frequently observed, while others are encountered less often.

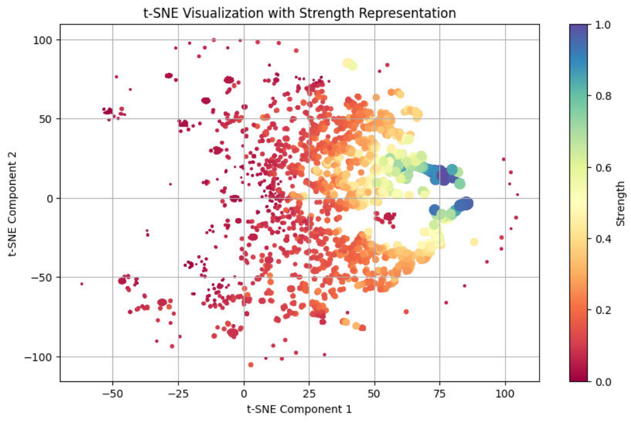

FIGURE 6. T-SNE visualization of state distribution observed during RL agent training. Larger points with higher strengths represent frequently visited states.

The t-SNE plot in Figure 6 provides a visual representation of the distribution of states as observed by the RL agent during training. This visualization transforms complex agent observations into a map of points, with each point representing a state visited by the agent. The size and strength of these points increase as the agent frequents those states. In Figure 6, certain points that are larger, represent states with obstacle-free paths, and their presence in dense clusters in the plot signifies the agent's preference for them throughout the training process. In contrast, states that are less frequently visited, and are likely associated with obstacles, are shown as sparse and spread-out clusters of smaller points. This observation implies that, due to the agent's preference for certain states, numerous states remain under-explored. Consequently, this leads to imbalanced training data, which, in turn, results in skewed outcomes in the navigation policy.

To address the imbalance in the exposed states within the training dataset, we use a heuristic approach to assign sample weights to different states based on their occurrences in the training set. If S is the set of states in the training batch with a total of n states ($|S| = n$), and number of possible actions a, the weight w_i for each state s_i in S is calculated as shown in equation 9.

$$w_i = \frac{1}{c_i} \times \frac{n}{a} \tag{9}$$

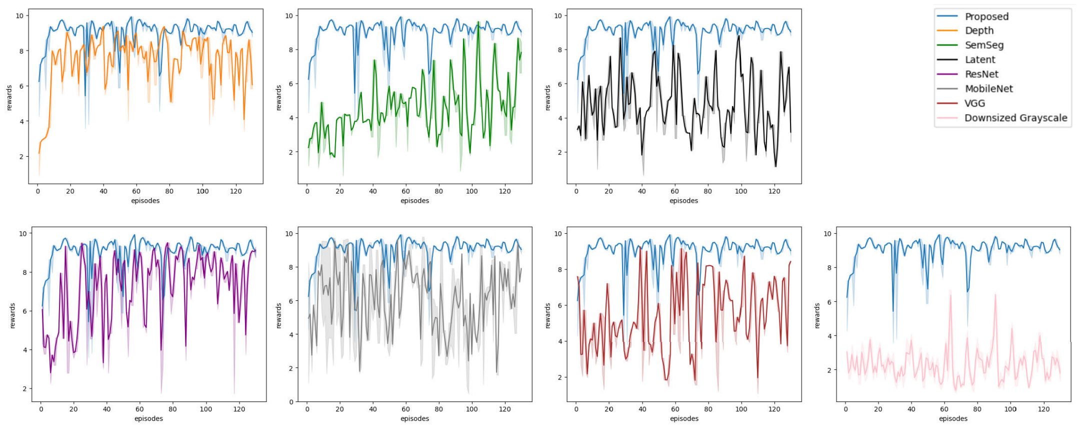

FIGURE 7. Learning curve of the RL model using different state representations.

where c_i is the count of occurrences of s_i in S. By introducing these weights, the navigation policy can now take into account the under-represented states, leading to more balanced and effective training outcomes.

C. TRAINING SETUP

The reinforcement learning agent in this study underwent training on randomly generated episodes within the AI2-Thor environment. Each episode was restricted to a length of 150 steps to accommodate the limited exploration space within each scene. The AI2-Thor environment comprises 120 scenes, divided into four distinct categories. 80% of the scenes, 20% from each category are used for training and the remaining scenes are used for testing the agent. To ensure maximum exploration of the environment, the agent was made to navigate each scene five times, each with a different starting point and orientation. This enabled the agent to traverse the environment from different viewpoints and take varied walks in the same environment. The agent perceived the environment through a first-person camera view of a 300×300 RGB image, and its action space consisted of five distinct actions, namely MoveLeft, MoveRight, MoveAhead, TurnLeft, and TurnRight. The model was trained to navigate safely within the AI2-Thor environment by utilizing state information along with a carefully designed reward system.

D. BASELINE AND EVALUATION METRICS

In this study, we evaluate the performance of reinforcement learning algorithms using our proposed approach for state representation, and compare it with alternative state spaces such as plain semantic segmentation image, depth image, as well as state as input features extracted using state-of-the-art techniques such as Downsized Grayscale image [1], ResNet [39], MobileNet [40], VGG16 [41] and latent representation generated using Autoencoders [4]. The models are trained on 22 scenes of each category provided by Ai2-thor. The depth and semantic segmentation image modalities provided by the framework are used for constructing the corresponding baseline models for depth and semantic segmentation states. The proposed compact state representation is derived from the semantic segmentation image, as described in Algorithm 1.

The learning curve of the agent during the training process is presented in Fig. 7. The rewards experienced by the agent increase over the course of training episodes and gradually converge to a stable value. Notably, the use of state input from the proposed SAT yields a faster and stable learning curve compared to alternative techniques. State information in the form of Depth data and ResNet features also converge and stabilize faster when compared to the other techniques.

To assess the performance of RL using different state representations, we use rate of Collisions, Movements and Rewards as the desirable metrics. Collision rate measures the percentage of trials where the agent collides with an obstacle. A lower collision rate indicates a safer and more effective navigation strategy. The rich sensory data received after every interaction of the agent in AI2-Thor environment can be used to detect collisions in the environment. The computation of collision rate is shown in (10).

$$C = \frac{N_c}{N} \tag{10}$$

where N is the total number of steps taken and N_c is the number of steps failed due to collision with objects.

An ideal navigation assistance solution facilitates safe navigation with minimal navigation instructions. Too many Move left, Move right, Turn left, Turn right orders might be unpleasant during navigation. Any navigation command other than Move Ahead is accounted for movement rate as shown in (11).

$$M = \frac{M_{total} - M_{ahead}}{M_{total}} \tag{11}$$

where M_{total} is the total number of movement commands given, and M_{ahead} is the number of MoveAhead commands issued during an episode.

TABLE 1. Comparison of system performance across different scenes of AI2-Thor environment.

Method	Kitchen			Living room			Bed room			Bath room		
	C	M	R	C	M	R	C	M	R	C	M	R
Semseg	55.60	55.60	42.14	48.80	48.80	48.81	49.50	49.50	46.18	60.60	71.25	41.85
Depth	9.95	34.20	48.19	15.00	30.05	49.08	7.25	28.60	62.53	9.05	21.95	66.08
Latent	23.3	32.25	74.03	24.15	25.45	75.14	20.1	19.75	81.385	20.9	18.90	81.32
Downsized Grayscale	10.25	0.00	89.75	12.85	0.00	87.15	10.85	0.00	89.75	9.4	0.00	90.60
MobileNet	12.05	54.8	45.69	15.45	49.3	47.34	13.45	38.9	55.84	2.25	82.85	33.13
VGG	3.00	20.80	80.96	6.65	19.15	78.46	29.45	17.9	56.45	8.05	29.15	69.00
ResNet	1.95	36.75	68.65	10.45	38.15	59.04	6.55	34.25	66.18	0.33	56.46	54.50
Proposed	**0.35**	**14.20**	**88.48**	**1.85**	**18.35**	**83.94**	**0.45**	**15.75**	**86.20**	**0.40**	**13.50**	**88.75**

Rewards refers to the *reward* returned by the agent while navigating in the environment as shown in (6). Higher rewards correspond to navigation in obstacle free spaces.

VI. RESULTS AND DISCUSSION

We carry out experiments on the proposed RL-based visual navigation and assess the performance of the proposed system in comparison to baseline methods. In the virtual environment testing phase, the agent navigates each scene 5 times with random start positions and orientations. The performance is then averaged over the 5 runs for each scene. Table 1 shows the performance of the agents across different scene categories of AI2-Thor environment.

Our findings show that the proposed system performs optimally across different scene types, with lower collision and movement rates and higher rewards when compared to the usage of image features extracted using state-of-the-art techniques. While the remaining methods performed fairly well on the scenes in the training set, their performance decreased drastically in the unseen scenes of test dataset. The model trained using downsized grayscale states exhibited minimal learning, as it consistently generated "Moveahead" instructions regardless of the available state information. While downsized grayscale images could have proven effective in static industrial insertion tasks [1], the substantial variations in the training data collected by a mobile agent are likely to have hindered the agent's learning capacity, resulting in an underfitting scenario. The proposed method of state extraction is able to generalize the state information across different scene categories. A model that learns from this abstract state representation is robust to changes in scene types and also adapts itself seamlessly in new and unseen virtual and real environments.

The percentage improvement in performance was computed as

$$\frac{Current\_best - Proposed}{Current\_best} \tag{12}$$

The proposed method provided 84.18% fewer collisions, 28.96% fewer movement instructions and 11.3% higher rewards compared to the best alternative options available.

Qualitative Analysis of Proposed System: Table 2 presents a qualitative analysis of the proposed approach compared to the baseline models. The analysis focuses on several important aspects of the models, including model size, number of trainable parameters, and inference speed, both in the original form and in the tflite format (optimized for mobile deployment).

TABLE 2. Qualitative analysis of proposed approach.

Input Mode	Model Size	Trainable parameters	Inference speed	Model Size	Inference speed
Semseg	207.4 MB	17,284,549	0.087 s	65.9 MB	5.4e-03 s
Depth	69.2 MB	5,764,549	0.078 s	22.0 MB	1.6e-03 s
Latent	243 kB	100,229	0.079 s	403.0 kB	3.0e-05 s
Downsized	874 kB	70,085	0.088 s	276 kB	3.9e-05 s
ResNet	1.5 MB	135,621	0.105 s	531.8 kB	4.5e-05 s
MobileNet	853 kB	70,085	0.072 s	275.4 kB	2.6e-05 s
VGG	162 kB	37,317	0.073 s	146.6 kB	1.8e-05 s
Proposed	**120 kB**	**7621**	**0.046 s**	**32.4 kB**	**1.2e-05 s**

The proposed approach has the smallest model size (in both original and tflite formats) which is significantly smaller than the alternatives, making it more suitable for resource-constrained environments, such as mobile devices or IoT devices, where limited storage is a concern. The proposed approach also has the fewest trainable parameters compared to other models. This suggests that the proposed system has a simpler and more compact model architecture resulting in faster training and reduced risk of overfitting. The inference speed of the proposed system is faster than the other models and is even faster in tflite format. In conclusion, the proposed approach shows promising results in terms of its smaller model size, fewer trainable parameters, and faster inference speed compared to the other models. These characteristics make it a strong candidate for resource-constrained environments and real-time applications.

A. ABLATION STUDIES

In this section, we discuss the ablation experiments performed to evaluate the contributions of each component of the proposed approach. Specifically we conduct experiments to understand the significance of State Abstraction Technique and the Reward module. DDQN is used as a common reinforcement learning algorithm for all the (state + reward) combinations given below:

- Semantic Segmentation(semseg) + action rewards (R_a) - In this setup, semantic segmentation of the captured

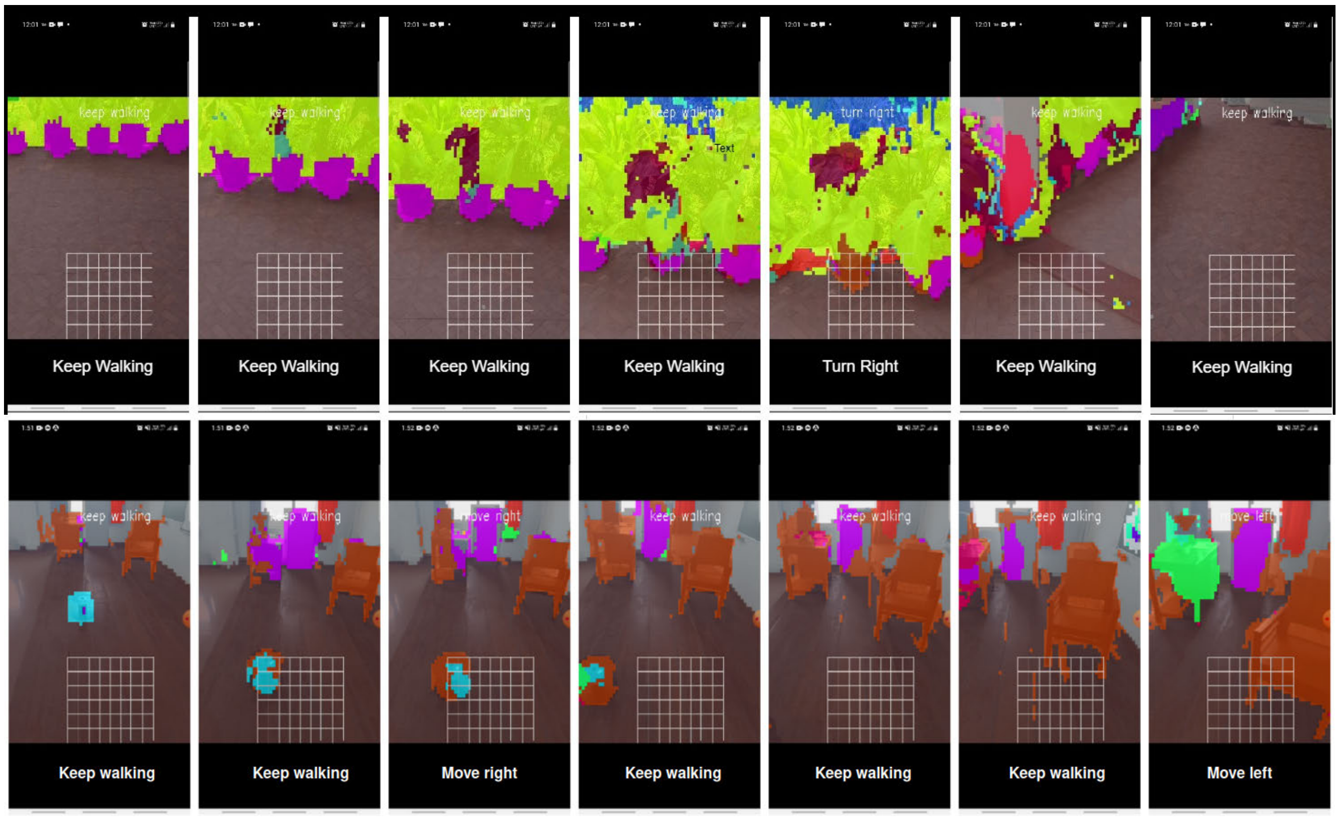

FIGURE 8. Deployment of proposed system in the real world. Outdoor (top row) Indoors (bottom row).

image is given as input to the RL model. A reward setup as described in (4) is used.

- Proposed SAT + no rewards - The proposed state abstraction method is used to represent the state information. The associated reward module is removed from the setup.
- Proposed SAT + action rewards (R_a) - The proposed state abstraction method along with action rewards is used in this setup.
- Proposed SAT + state grid rewards (R_g) - The proposed state abstraction method along with state grid rewards as described in (3) is used in this setup.
- Proposed SAT + state grid rewards (R_g) + action rewards (R_a) - The complete proposed system with the state abstraction and reward modules are used in this setup.

Table 3 reflects the significance of the proposed SAT and Reward modules. The results of the study showed that the proposed state abstraction technique is a critical component of the approach. Upgrading the state representation from semantic segmentation to the proposed state abstraction technique resulted in a significant reduction in the collision rate. This indicates that the proposed state abstraction technique allows the RL model to better adapt to new and unknown environments, resulting in a better performance.

TABLE 3. Ablation study on proposed system in terms of collision rates, number of movements and rewards.

Method	Collision	Movements	Rewards
Semseg+R_a	41.93	65.97	44.95
SAT only	5.54	95.02	47.50
SAT + R_a	15.66	**8.16**	82.05
SAT + R_g	5.48	22.33	86.41
SAT + R_a+R_g	**0.76**	15.45	**86.41**

The reward shaping components, R_a and R_g, were also found to be crucial to the overall performance of the approach. The movement rates and rewards are mostly governed by reward shaping. R_a regulates the movement commands, whereas R_g is concerned with collision avoidance. Therefore, the movement and reward rates, which were sub-optimal with SAT-only, were seen to be improved with the introduction of reward shaping components R_a and R_g. The results show that the proposed approach's performance was significantly improved when both R_a and R_g were used with the proposed state abstraction technique.

In conclusion, the ablation study highlights the importance of the proposed state abstraction technique and reward modules in the proposed approach's overall performance. The results suggest that the proposed approach can be further improved by optimizing these components to achieve better performance in different scenarios.

TABLE 4. Execution time of the modules in the proposed architecture.

Module	Execution Time (s)	Contribution
Perception Module	**0.06**	99.93%
State Abstraction	2.38e-07	0.0003%
Reward Module	3.40e-05	0.056%
RL Module	1.2e-05	0.009%
Total	**0.06 seconds**	

B. SYSTEM PERFORMANCE

The execution cost of the proposed system's component modules is shown in Table 4. The total response time of the system is 0.06 seconds. As such the system is highly responsive and is able to provide feedback about obstacles in near-real time. The major contributor of the system response time is the semantic segmentation, that accounts for up to 99% of the system response time. This indicates that faster segmentation models in the future will further improve the performance of the system.

C. REAL WORLD DEPLOYMENT

Using semantic segmentation representation instead of RGB image has the benefit of concealing the dissimilarities in observations between real and virtual environments. The proposed method involves processing the semantic segmentation image to generate an abstract representation of the state, thereby further enhancing the model's generalization capabilities. The reinforcement learning model was trained and tested on the Ai2-thor environment, and eventually deployed as a tflite model on an Android application for real-world experience. While it is noteworthy that the semantic segmentation produced by real-world models may not be as precise as those in the Ai2-thor environment, the abstract representation of the state can still be extracted effectively from these imperfect semantic segment images to produce reliable navigation guidance as illustrated in Fig. 8.

The proposed system was deployed on a smartphone to test its usability in the real world. The smartphone chosen for the research is the Samsung Galaxy A30s running on Android v11 Operating System with a Exynos 7904 Octa-core processor, 3 GB RAM, 25MP main camera, and 3808mV Li-ion battery. Fig. 8 illustrates the effectiveness of the proposed method in providing navigation guidance in the real world. The figure shows that the model can successfully navigate through complex real-world environments with narrow spaces and obstacles. The proposed method's ability to provide reliable navigation guidance in real-world scenarios demonstrates its potential for deployment in various applications, such as autonomous vehicles, robot navigation, navigation assistance for the vision impaired, etc.

VII. CONCLUSION

The proposed study enhances RL based visual navigation in resource constrained settings by introducing a State Abstraction Technique that brings with it the two-fold advantage of Dimensionality Reduction and Sim2Real transferability. The reduced state space helps in faster and efficient learning of navigation policies using simple RL models. A key aspect of our approach is the usage of motion blurred pictures which ensures that the system is designed with real-world deployment in mind. Our approach demonstrates significant improvement in visual navigation and collision avoidance when compared to policy learning done using other state-of-the-art state feature extraction techniques. The demonstrated improvements highlight the potential impact of SAT in enabling more efficient and reliable navigation in real-world scenarios. The generic and environment-independent state information generated by SAT further enhances the transferability of our approach.

One aspect to consider in the proposed SAT approach is that it relies on the current state of the environment for collision detection and avoidance. In scenarios where objects are moving towards the agent at higher speeds, the proposed system may not be equipped to detect and notify events on time to avoid collisions. To address this limitation, future work could involve the addition of an anticipation component to SAT, which would anticipate the future positions of moving objects when performing collision detection and avoidance. By incorporating predictions of object motion, SAT could better adapt to dynamic environments and enable more proactive collision avoidance strategies.

REFERENCES

[1] G. Schoettler, A. Nair, J. Luo, S. Bahl, J. Aparicio Ojea, E. Solowjow, and S. Levine, "Deep reinforcement learning for industrial insertion tasks with visual inputs and natural rewards," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2020, pp. 5548–5555.

[2] B. Eysenbach, R. R. Salakhutdinov, and S. Levine, "Search on the replay buffer: Bridging planning and reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 1–12.

[3] F. Raudies, S. Eldridge, A. Joshi, and M. Versace, "Learning to navigate in a virtual world using optic flow and stereo disparity signals," *Artif. Life Robot.*, vol. 19, no. 2, pp. 157–169, Sep. 2014.

[4] C. Huang, R. Zhang, M. Ouyang, P. Wei, J. Lin, J. Su, and L. Lin, "Deductive reinforcement learning for visual autonomous urban driving navigation," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 12, pp. 5379–5391, Dec. 2021.

[5] R. Shah and V. Kumar, "RRL: ResNet as representation for reinforcement learning," 2021, *arXiv:2107.03380*.

[6] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2017, pp. 23–30.

[7] A. A. Rusu, S. G. Colmenarejo, C. Gulcehre, G. Desjardins, J. Kirkpatrick, R. Pascanu, V. Mnih, K. Kavukcuoglu, and R. Hadsell, "Policy distillation," 2015, *arXiv:1511.06295*.

[8] X. Peng, R. Chen, J. Zhang, B. Chen, H.-W. Tseng, T.-L. Wu, and T.-H. Meen, "Enhanced autonomous navigation of robots by deep reinforcement learning algorithm with multistep method," *Sensors Mater.*, vol. 33, no. 2, pp. 825–842, 2021.

[9] L. Sun, J. Zhai, and W. Qin, "Crowd navigation in an unknown and dynamic environment based on deep reinforcement learning," *IEEE Access*, vol. 7, pp. 109544–109554, 2019.

[10] Q. Wu, K. Xu, J. Wang, M. Xu, X. Gong, and D. Manocha, "Reinforcement learning-based visual navigation with information-theoretic regularization," *IEEE Robot. Autom. Lett.*, vol. 6, no. 2, pp. 731–738, Apr. 2021.

[11] A. Singla, S. Padakandla, and S. Bhatnagar, "Memory-based deep reinforcement learning for obstacle avoidance in UAV with limited environment knowledge," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 1, pp. 107–118, Jan. 2021.

[12] K. Zhou, C. Guo, and H. Zhang, "Visual navigation via reinforcement learning and relational reasoning," in *Proc. IEEE SmartWorld, Ubiquitous Intell. Comput., Adv. Trusted Comput., Scalable Comput. Commun., Internet People Smart City Innov. (SmartWorld/SCALCOM/UIC/ATC/IOP/SCI)*, Oct. 2021, pp. 131–138.

[13] A. Rafiei, A. O. Fasakhodi, and F. Hajati, "Pedestrian collision avoidance using deep reinforcement learning," *Int. J. Automot. Technol.*, vol. 23, no. 3, pp. 613–622, Jun. 2022.

[14] C. Ye, N. H. Yung, and D. Wang, "A fuzzy controller with supervised learning assisted reinforcement learning algorithm for obstacle avoidance," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 33, no. 1, pp. 17–27, Feb. 2003.

[15] Z. Rao, Y. Wu, Z. Yang, W. Zhang, S. Lu, W. Lu, and Z. Zha, "Visual navigation with multiple goals based on deep reinforcement learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 12, pp. 5445–5455, Dec. 2021.

[16] Y. Lu, Y. Chen, D. Zhao, and D. Li, "MGRL: Graph neural network based inference in a Markov network with reinforcement learning for visual navigation," *Neurocomputing*, vol. 421, pp. 140–150, Jan. 2021.

[17] J. Kulhánek, E. Derner, T. de Bruin, and R. Babuška, "Vision-based navigation using deep reinforcement learning," in *Proc. Eur. Conf. Mobile Robots (ECMR)*, Sep. 2019, pp. 1–8.

[18] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi, "Target-driven visual navigation in indoor scenes using deep reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2017, pp. 3357–3364.

[19] D. Corneil, W. Gerstner, and J. Brea, "Efficient model-based deep reinforcement learning with variational state tabulation," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1049–1058.

[20] X. Huang, W. Chen, W. Zhang, R. Song, J. Cheng, and Y. Li, "Autonomous multi-view navigation via deep reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2021, pp. 13798–13804.

[21] Z. Liu, Y. Cao, J. Chen, and J. Li, "A hierarchical reinforcement learning algorithm based on attention mechanism for UAV autonomous navigation," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 11, pp. 13309–13320, Nov. 2023.

[22] H. Zhu, J. Yu, A. Gupta, D. Shah, K. Hartikainen, A. Singh, V. Kumar, and S. Levine, "The ingredients of real-world robotic reinforcement learning," 2020, *arXiv:2004.12570*.

[23] Y. Wu, Y. Wu, A. Tamar, S. Russell, G. Gkioxari, and Y. Tian, "Bayesian relational memory for semantic visual navigation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 2769–2779.

[24] A. Devo, G. Mezzetti, G. Costante, M. L. Fravolini, and P. Valigi, "Towards generalization in target-driven visual navigation by using deep reinforcement learning," *IEEE Trans. Robot.*, vol. 36, no. 5, pp. 1546–1561, Oct. 2020.

[25] X. Huang, H. Deng, W. Zhang, R. Song, and Y. Li, "Towards multi-modal perception-based navigation: A deep reinforcement learning method," *IEEE Robot. Autom. Lett.*, vol. 6, no. 3, pp. 4986–4993, Jul. 2021.

[26] Z.-W. Hong, C. Yu-Ming, S.-Y. Su, T.-Y. Shann, Y.-H. Chang, H.-K. Yang, B. H.-L. Ho, C.-C. Tu, Y.-C. Chang, T.-C. Hsiao, H.-W. Hsiao, S.-P. Lai, and C.-Y. Lee, "Virtual-to-real: Learning to control in visual semantic segmentation," 2018, *arXiv:1802.00285*.

[27] H.-C. Chen, "Monocular vision-based obstacle detection and avoidance for a multicopter," *IEEE Access*, vol. 7, pp. 167869–167883, 2019.

[28] S. Vorapatratorn, "AI-based obstacle detection and navigation for the blind using convolutional neural network," in *Proc. 25th Int. Comput. Sci. Eng. Conf. (ICSEC)*, Nov. 2021, pp. 17–22.

[29] V.-N. Hoang, T.-H. Nguyen, T.-L. Le, T.-H. Tran, T.-P. Vuong, and N. Vuillerme, "Obstacle detection and warning system for visually impaired people based on electrode matrix and mobile Kinect," *Vietnam J. Comput. Sci.*, vol. 4, no. 2, pp. 71–83, May 2017.

[30] G. Dimas, D. E. Diamantis, P. Kalozoumis, and D. K. Iakovidis, "Uncertainty-aware visual perception system for outdoor navigation of the visually challenged," *Sensors*, vol. 20, no. 8, p. 2385, Apr. 2020.

[31] M. Hua, Y. Nan, and S. Lian, "Small obstacle avoidance based on RGB-D semantic segmentation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. Workshop (ICCVW)*, Oct. 2019, pp. 886–894.

[32] L. Sun, K. Yang, X. Hu, W. Hu, and K. Wang, "Real-time fusion network for RGB-D semantic segmentation incorporating unexpected obstacle detection for road-driving images," *IEEE Robot. Autom. Lett.*, vol. 5, no. 4, pp. 5558–5565, Oct. 2020.

[33] P.-J. Duh, Y.-C. Sung, L. F. Chiang, Y.-J. Chang, and K.-W. Chen, "V-eye: A vision-based navigation system for the visually impaired," *IEEE Trans. Multimedia*, vol. 23, pp. 1567–1580, 2021.

[34] P. Wenzel, T. Schön, L. Leal-Taixé, and D. Cremers, "Vision-based mobile robotics obstacle avoidance with deep reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2021, pp. 14360–14366.

[35] K. Wu, H. Wang, M. A. Esfahani, and S. Yuan, "BND*-DDQN: Learn to steer autonomously through deep reinforcement learning," *IEEE Trans. Cogn. Develop. Syst.*, vol. 13, no. 2, pp. 249–261, Jun. 2021.

[36] K. Wu, H. Wang, M. A. Esfahani, and S. Yuan, "Learn to navigate autonomously through deep reinforcement learning," *IEEE Trans. Ind. Electron.*, vol. 69, no. 5, pp. 5342–5352, May 2022.

[37] E. Kolve, R. Mottaghi, W. Han, E. VanderBilt, L. Weihs, A. Herrasti, M. Deitke, K. Ehsani, D. Gordon, Y. Zhu, A. Kembhavi, A. Gupta, and A. Farhadi, "AI2-THOR: An interactive 3D environment for visual AI," 2017, *arXiv:1712.05474*.

[38] W. Zhang, Z. Huang, G. Luo, T. Chen, X. Wang, W. Liu, G. Yu, and C. Shen, "TopFormer: Token pyramid transformer for mobile semantic segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 12083–12093.

[39] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[40] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient convolutional neural networks for mobile vision applications," 2017, *arXiv:1704.04861*.

[41] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*.

U. VIJETHA (Graduate Student Member, IEEE) received the B.E. degree in information science and the M.Tech. degree in computer science from Visvesvaraya Technological University, Karnataka, India. She is currently a Ph.D. Research Scholar with the Department of Information Technology, National Institute of Technology Karnataka (NITK), Surathkal. She was a Faculty Member with the Department of Computer Science and Engineering, St. Joseph Engineering College, Mangaluru, Karnataka, from 2012 to 2020. In 2020, she has joined as a full-time Ph.D. Research Scholar with the Department of Information Technology, NITK. Her research interests include image processing, computer vision, and machine learning.

V. GEETHA (Senior Member, IEEE) received the B.E. degree in computer science and engineering from Mangalore University, Karnataka, India, the M.Tech. degree in computer science and engineering from Visvesvaraya Technological University (VTU), Belagavi, India, and the Ph.D. degree in computer science and engineering from the National Institute of Technology Karnataka (NITK), Surathkal, Karnataka. From 2000 to 2008, she was a Faculty Member with the Department of Computer Science and Engineering, NMAMIT, Nitte, Karkala, India. Since 2008, she has been an Assistant Professor with the Information Technology Department, NITK. Her research interests include wireless sensor networks, the Internet of Things, and parallel computing. She has authored more than 30 publications in international conferences and journals. Her awards and honors include the Young Faculty Research Fellow Award 2019–2021, under Ph.D. Scheme of Ministry of Electronics & Information Technology (MeitY), Government of India's Digital Corporation (Formerly Media Laboratory Asia), and Second Rank in M.Tech., VTU.